From: А.С.Самотохин

Subj: Описание Формата GIF

G I F (tm)

Graphics Interchange Format (tm)

(Формат графического обмена)

Стандартное определение механизма

для запоминания и передачи

растровой графической информации

June 15, 1987

(c) CompuServe Incorporated, 1987

All rights reserved

При копировании данного документа

содержащаяся в нем информация становится

доступна для пользователей компьютеров без

лицензионных ограничений

GIF и 'Graphics Interchange Format'

являются торговой маркой CompuServe, Incorporated.

an H&R Block Company 5000 Arlington Centre Blvd.

Columbus, Ohio 43220

(614) 457-8600

Перевод с английского языка выполнен А.С.Самотохиным

Институт прикладной математики АН СССР

Москва, ноябрь 1990 г.

- 2 -

Описание формата графического обмена (GIF)

Оглавление

ВВЕДЕНИЕ . . . . . . . . . . . . . . . . . . . 3

ОБЩИЙ ФОРМАТ ФАЙЛА . . . . . . . . . . . . . . 3

ИДЕНТИФИКАТОР GIF. . . . . . . . . . . . . . . 4

ДЕСКРИПТОР ЭКРАНА. . . . . . . . . . . . . . . 4

ГЛОБАЛЬНАЯ ТАБЛИЦА ЦВЕТОВ. . . . . . . . . . . 5

ДЕСКРИПТОР ИЗОБРАЖЕНИЯ . . . . . . . . . . . . 6

ЛОКАЛЬНАЯ ТАБЛИЦА ЦВЕТОВ . . . . . . . . . . . 7

РАСТРОВЫЕ ДАННЫЕ . . . . . . . . . . . . . . . 7

ТЕРМИНАТОР GIF . . . . . . . . . . . . . . . . 8

РАСШИРЕННЫЙ БЛОК GIF . . . . . . . . . . . . . 8

ПРИЛОЖЕНИЕ A - ГЛОССАРИЙ . . . . . . . . . . . 9

ПРИЛОЖЕНИЕ B - ВЗАИМОДЕЙСТВУЮЩИЕ

ПОСЛЕДОВАТЕЛЬНОСТИ. . . . . . . 10

ПРИЛОЖЕНИЕ C - УПАКОВКА И СЖАТИЕ ИЗОБРАЖЕНИЯ . 12

ПРИЛОЖЕНИЕ D - ОБРАБОТКА НЕСКОЛЬКИХ

ИЗОБРАЖЕНИЙ . . . . . . . . . . 15

- 3 -

ВВЕДЕНИЕ

'GIF' (tm) - это стандарт фирмы CompuServe для определения

растровых цветных изображений. Этот формат позволяет высвечивать

на различном оборудовании графические высококачественные

изображения с большим разрешением и подразумевает механизм обмена

и высвечивания изображений. Описанный в настоящем документе формат

изображений был разработан для поддержки настоящей и будущей

технологии обработки изображений и будет в дальнейшем служить

основой для будущих графических продуктов CompuServe.

Главная задача настоящего документа состоит в том, чтобы

снабдить программистов необходимой технической информацией для

написания декодеров и кодировщиков GIF. Поэтому в документе

используется терминология связанная с общими вопросами графики и

программирования.

Первый раздел настоящего документа описывает формат данных

GIF и его компоненты в приложении к декодерам GIF, вне зависимости

от того являются ли они отдельной программой или частью пакета

связи. Приложение B относится к декодерам являющимися частью

пакетов связи и описывает протокол, необходимый для входа и

существования режима GIF и отвечает на ряд специфических вопросов.

Глоссарий в приложении A определяет некоторые термины,

использованные в документе. Приложение C дает подробное объяснение

того, как сами графические изображения пакуются в виде

последовательности байтов.

Определение формата данных GIF

ОБЩИЙ ФОРМАТ ФАЙЛА

┌───────────────────────────────┐

│ ┌───────────────────────────┐ │

│ │ Идентификатор GIF │ │

│ └───────────────────────────┘ │

│ ┌───────────────────────────┐ │

│ │ Дескриптор экрана │ │

│ └───────────────────────────┘ │

│ ┌───────────────────────────┐ │

│ │ Глобальная таблица цветов │ │

│ └───────────────────────────┘ │

. . . . . .

│ ┌───────────────────────────┐ │ ───┐

│ │ Дескирптор изображения │ │ │

│ └───────────────────────────┘ │ │

│ ┌───────────────────────────┐ │ │

│ │ Локальная таблица цветов │ │ ├── Повторяется

│ └───────────────────────────┘ │ │ от 1 до n раз

│ ┌───────────────────────────┐ │ │

│ │ Растровые данные │ │ │

│ └───────────────────────────┘ │ ───┘

. . . . . .

├─ Терминатор GIF ─┤

└───────────────────────────────┘

- 4 -

ИДЕНТИФИКАТОР GIF

Наличие в начале файла специальной "подписи" указывает, что

последующие данные являются действительно потоком данных

изображения в формате GIF. Эта "подпись" состоит из следующих

шести символов:

G I F 8 7 a

Три последних символа '87a' могут рассматриваться как номер версии

для данного конкретного определения GIF и будут использоваться в

дальнейшем в качестве ссылки на документ с описанием GIF в

зависимости от номера версии.

ДЕСКРИПТОР ЭКРАНА

Дескриптор экрана описывает общие параметры для всех

последующих изображений в формате GIF. Он определяет размеры

пространства изображения или требуемого логического экрана,

существование информации о таблице цветов и "глубине" экрана. Эта

информация запоминается в виде серии 8-битовых байтов, как

показано ниже.

биты

7 6 5 4 3 2 1 0 Номер байта

┌───────────────┐

│ │ 1

├─Ширина экрана─┤ Ширина растра в пикселах (сначала LSB)

│ │ 2

├───────────────┤

│ │ 3

├─Высота экрана─┤ Высота растра в пикселах (сначала LSB)

│ │ 4

├─┬─────┬─┬─────┤ M = 1, За дескриптором следует глобальная

│ │ │ │ │ таблица цветов

│M│ cr │0│pixel│ 5 cr+1 = число битов цветового разрешения

├─┴─────┴─┴─────┤ pixel+1 = число бит/пиксел в изображении

│ background │ 6 фон = цветовой индекс фона экрана

├───────────────┤ (цвет определяется из глобальной таблицы

│0 0 0 0 0 0 0 0│ 7 цветов или из таблицы по умолчанию)

└───────────────┘

Ширина и высота логического экрана могут быть больше размеров

физического экрана. Способ высвечивания изображений больших, чем

размеры физического экрана зависит от реализации и может

использовать преимущества конкретного оборудования (например, окна

скроллинга в Macintosh scrolling windows). В противном случае

изображение будет усечено по краям экрана.

Значение 'pixel' также определяет число цветов в изображении.

Диапазон значений 'pixel' составляет от 0 до 7, что соответствует

от 1 до 8 битам. Это транслируется в диапазон от 2 (черно-белые

изображения) до 256 цветов. Бит 3 в байте 5 зарезервирован для

будущих определений и должен быть нулевым.

- 5 -

ГЛОБАЛЬНАЯ ТАБЛИЦА ЦВЕТОВ

Глобальная таблица цветов является необязательной и

рекомендуется для изображений, где требуется точная передача

цветов. На существование этой таблицы указывает поле 'M' в байте 5

дескриптора экрана. Цветовая таблица может быть также связана с

каждым изображением в GIF-файле, что будет описано позже. Однако

обычно эта глобальная таблица будет использоваться, из-за

ограничений, существующих в настоящее время в доступном

оборудовании. Флаг 'M' в дескрипторе конкретного изображения

обычно равен 0. Если глобальная таблица цветов присутствует, ее

определение следует непосредственно за дескриптором экрана. Число

элементов цветовой таблицы, следующей за описателем экрана равно

2\*\*(число бит/пиксел), причем каждый элемент состоит из трех

байтов, значения которых описывают соответственно относительную

интенсивность красного, зеленого и синего цветов. Структура блока

цветовой таблицы:

биты

7 6 5 4 3 2 1 0 Байт #

┌───────────────┐

│интен. красного│ 1 Значение красного для цвета 0

├───────────────┤

│интен. зеленого│ 2 Значение зеленого для цвета 0

├───────────────┤

│интен. синего │ 3 Значение синего для цвета 0

├───────────────┤

│интен. красного│ 4 Значение красного для цвета 1

├───────────────┤

│интен. зеленого│ 5 Значение зеленого для цвета 1

├───────────────┤

│интен. синего │ 6 Значение синего для цвета 1

├───────────────┤

: : (Продолжение для остальных цветов)

Получаемое значение каждого пиксела при высвечивании

изображения будет соответствовать ближайшему доступному цвету из

цветовой таблицы дисплея. Цветовые компоненты представляют собой

значение относительной интенсивности от нулевой (0) до полной

(255). Белый цвет может быть представлен как (255,255,255), черный

как (0,0,0) и желтый как (180,180,0). При высвечивании на

дисплеях, которые поддерживают менее 8 бит на цветовую компоненту,

используются старшие биты. При создании элементов цветовой

таблицы GIF на аппаратуре, поддерживающей менее 8 бит на

компоненту, значение аппаратной компоненты должно быть

конвертировано в 8-битный формат по следующей формуле:

<значение\_в\_таблице> = <компонента>\*255/(2\*\*<число\_бит> -1)

Это обеспечивает точный перевод цветов для всех дисплеев. В

случае создания изображения GIF на аппаратуре без возможности

цветовой палитры, должна быть создана фиксированная палитра на

основе доступных для данного оборудования цветов. Если указано

отсутствие глобальной таблицы цветов, цветовая таблица по

умолчанию генерируется внутренним образом так, что каждый цветовой

индекс равен аппаратному цветовому индексу modulo <n>, где <n> -

число доступных цветов на оборудовании.

- 6 -

ДЕСКРИПТОР ИЗОБРАЖЕНИЯ

Дескриптор изображения определяет действительное расположение

и размеры последующего изображения внутри пространства,

определенного в дескрипторе экрана. Также определяются флаги,

указывающие на присутствие локальной таблицы для поиска цветов и

определения последовательности высвечивания пикселов. Каждый

дескриптор изображения начинается с символа-разделителя

изображений. Роль разделителя изображений состоит просто в

синхронизации при входе в дескриптор изображения. Это желательно,

если GIF-файл состоит более, чем из одного изображения. Этот

символ определен как шестнадцатиричное 0x2C или ',' (запятая). Как

только этот символ встречается между изображениями,

непосредственно за ним следует дескриптор изображения.

Любой символ, встреченный между концом предыдущего

изображения и символом-разделителем изображения игнорируется. Это

позволит при последующих модификациях GIF допускать присутствие

нескольких форматов и правильно игнорировать их старыми

декодерами.

биты

7 6 5 4 3 2 1 0 Байт #

┌───────────────┐

│0 0 1 0 1 1 0 0│ 1 ',' - Символ-разделитель изображения

├───────────────┤

│ │ 2 Начало изображения в пикселах относи-

├─ Левый край ─┤ тельно левого края экрана (сначала LSB)

│ │ 3 Least significant byte

├───────────────┤

│ │ 4

├─ Верхний край─┤ Начало изображения в пикселах относительно

│ │ 5 верхнего края экрана (сначала LSB)

├───────────────┤

│ │ 6

├─ Ширина ─┤ Ширина изображения в пикселах

│ │ 7 (сначала LSB)

├───────────────┤

│ │ 8

├─ Высота ─┤ Высота изображения в пикселах

│ │ 9 (сначала LSB)

├─┬─┬─┬─┬─┬─────┤ M=0 - Использовать глобальную таблицу цве-

│M│I│0│0│0│pixel│ 10 тов, игнорировать 'pixel'

└─┴─┴─┴─┴─┴─────┘ M=1 - Далее следует локальная таблица цве-

тов, использовать 'pixel'

I=0 - Изображение отформатировано в после-

довательном порядке

I=1 - Изображение отформатировано в поряд-

ке переплетения

pixel+1 - число бит на пиксел в данном

изображении

Описание положения и размеров экрана должно быть находиться

внутри матрицы, определенной в дескрипторе экрана. С другой

стороны, нет необходимости, чтобы изображение полностью заполняло

весь экран.

- 7 -

ЛОКАЛЬНАЯ ТАБЛИЦА ЦВЕТОВ

Локальная таблица цветов необязательна и определена здесь для

будущего использования. Если установлен бит 'M' байта 10 в

дескрипторе изображения, то вслед за дескриптором изображения

следует локальная таблица цветов, которая относится только к

последующему изображению. После обработки изображения цветовую

таблицу следует привести к той, которая была определена после

дескриптора экрана. Заметим, что поле 'pixel' байта 10 в

дескрипторе изображения используется только в том случае, если

указана локальная таблица цветов. Она определяет не только размер

пиксела (число битов в нем), но число элементов последующей

цветовой таблицы. Число битов на пиксел также следует

восстановить к тому значению, которое было определено в

дескрипторе экрана, после того, как закончится обработка

изображения.

РАСТРОВЫЕ ДАННЫЕ

Формат самого изображения определен как серия значений

номеров пикселов, которые образуют изображение. Пикселы

запоминаются слева направо последовательно по строкам изображения.

По умолчанию строки записываются последовательно, сверху вниз. В

том случае, если установлен бит 'I' в байте 10 дескриптора

изображения, то порядок строк при записи изображения соответствует

четырех проходному процессу. При первом проходе записывается

каждая 8-ая строка, начиная с верхней строки окна изображения. При

втором проходе записывается каждая 8-ая строка, начиная с пятой

строки сверху. На третьем проходе записывается каждая 4-ая строка,

начиная с третьей строки окна. Четвертый проход завершает

изображение, записывая каждую вторую строку, начиная со второй

строки с сверху. Ниже приведено графическое описание этого

процесса.

Изображение

Стр. Прох.1 Прох.2 Прох.3 Прох.4 Результат

──────────────────────────────────────────────────────

0 \*\*1a\*\* \*\*1a\*\*

1 \*\*4a\*\* \*\*4a\*\*

2 \*\*3a\*\* \*\*3a\*\*

3 \*\*4b\*\* \*\*4b\*\*

4 \*\*2a\*\* \*\*2a\*\*

5 \*\*4c\*\* \*\*4c\*\*

6 \*\*3b\*\* \*\*3b\*\*

7 \*\*4d\*\* \*\*4d\*\*

8 \*\*1b\*\* \*\*1b\*\*

9 \*\*4e\*\* \*\*4e\*\*

10 \*\*3c\*\* \*\*3c\*\*

11 \*\*4f\*\* \*\*4f\*\*

12 \*\*2b\*\* \*\*2b\*\*

. . .

Значения пикселов изображения обрабатываются как цветовые

индексы, указывающие на существующую таблицу цветов. В результате

получается цветовое значение из таблицы, которое реально

воспроизводится на экране. Эти серии цветовых индексов, число

которых равно ширине\_изображения\*высоту\_изображения, пропускаются

через поток данных изображения GIF по одному значению на пиксел,

сжимаются и упаковываются в соответствии с версией алгоритма

сжатия LZW, как это определено в Приложении C.

- 8 -

ТЕРМИНАТОР GIF

Для того, чтобы обеспечить синхронизацию с окончанием файла

изображения GIF, декодер GIF должен обрабатывать окончание режима

GIF по символу шестнадцатиричное 0x3B или ';', найденному после

окончания обработки изображения. По соглашению декодирующие

программы должны делать паузу и ждать действий, указывающих, что

пользователь готов к продолжению. Это может быть возврат каретки,

введенный с клавиатуры или щелчок кнопкой мыши. Для интерактивных

приложений эти действия пользователя должны быть переданы в ядро

программы как перевод каретки, для того, чтобы вычислительный

процесс мог продолжаться. Обычно декодирующая программа покидает

графический режим и возвращается к предыдущему процессу.

РАСШИРЕННЫЙ БЛОК GIF

Для того, чтобы обеспечить аккуратное расширение определения

GIF, необходим механизм для определения упаковки внутри потока

данных GIF. Указанное расширение было определено и документировано

CompuServe для того, чтобы предусмотреть управляемый способ

усовершенствований.

Расширенный блок GIF пакуется способом, похожим на тот,

который использовался для растровых данных, но не сжимается.

Основная структура блока:

7 6 5 4 3 2 1 0 Байт #

┌───────────────┐

│0 0 1 0 0 0 0 1│ 1 '!' - Идентификатор расширенного блока

├───────────────┤

│ функц. код │ 2 Расширенный функциональный код (0-255)

├───────────────┤ ───┐

│ байт-счетчик │ │

├───────────────┤ │

│ │ ├── Повторяется столько раз, сколько

│ функ. байты │ │ необходимо

│ данных │ │

├───────────────┤ ───┘

. . . . . .

├───────────────┤

│0 0 0 0 0 0 0 0│ нулевой байт-счетчик (терминатор блока)

└───────────────┘

Расширенный блок GIF может непосредственно предшествовать

дескриптору изображения или находиться перед терминатором GIF.

Все декодеры GIF должны быть способны распознавать

присутствие расширенного блока GIF и затем читать его, если они не

могут обработать функциональный код. Это гарантирует, что старые

декодеры смогут обрабатывать файлы изображений GIF в будущем, хотя

и без дополнительных функциональных возможностей.

- 9 -

ПРИЛОЖЕНИЕ A

ГЛОССАРИЙ

Пиксел - Наименьший элемент графического изображения. Обычно

соответствует отдельной точке на графическом экране. Разрешение

изображения обычно задается в пикселах. Например, одним из

довольно стандартных экранных графических форматов является 320

пикселов по горизонтали на 200 по вертикали. Каждый пиксел

может быть окрашен одним из нескольких цветов в зависимости от

возможностей графического оборудования.

Растр - горизонтальные уровни пикселов, представляющие одну строку

изображения. Типичный метод порождения изображения, поскольку

большинство образцов видеоборудования ориентировано на наиболее

эффективную работу именно таким образом.

LSB - Сокращение от Least Significant Byte ( младший по значению

байт). Ссылается на соглашение для двух байтов числового

значения, согласно которому младший по значению байт

предшествует более старшему. Такое соглашение типично для

микрокомпьютеров.

Таблица цветов - Список определений для каждого цвета,

используемый в изображениях GIF. Желаемые цвета конвертируются

в доступные цвета с помощью таблицы, причем по входным цветовым

индексам изображения образуются выходные цветовые индексы

оборудования. Если для изображения GIF указана таблица цветов,

то цвета выходных пикселов будут изменены на основе

используемого оборудования и его способности соответствовать

заданным цветам.

Переплетение - Метод высвечивания изображений GIF, при котором

совершаются несколько проходов с выводом разнесенных строк

растра, что дает возможность визуализации общего содержания

всего изображения до того, как обработаны все данные.

B Протокол - Свободно распространяемый протокол передачи файлов с

исправлением ошибок, разработанный CompuServe и реализованный в

продукте VIDTEX фирмы CompuServe. Такой механизм обнаружения

ошибок будет использован при передаче изображений GIF для

интерактивных приложений.

LZW - Совершенный алгоритм сжатия данных, основанный на работе,

сделанной Lempel-Ziv и Welch, который обеспечивает возможность

высокоэффективного однопроходного кодирования и декодирования.

Это позволяет одновременно раскрывать и высвечивать

изображения. Исходная статья, в которой был описан указанный

метод:

Terry A. Welch, "A Technique for High

Performance Data Compression", IEEE Computer, vol 17

no 6 (June 1984)

Этот базовый алгоритм также используется в свободно

распространяемых утилитах ARC для сжатия файлов. Адаптация

алгоритма LZW, выполненная CompuServe для GIF описана в

приложении C.

- 10 -

ПРИЛОЖЕНИЕ B

ПОСЛЕДОВАТЕЛЬНОСТЬ ОБМЕНОВ GIF ДЛЯ ИНТЕРАКТИВНОЙ СРЕДЫ

Для управления на интерактивной линии связи между

отправителем и получателем GIF определена следующая

последовательность действий. Эта последовательность не применяется

в приложениях, включающих загрузку статических GIF-файлов и не

является частью GIF-файлов.

ЗАПРОС ВОЗМОЖНОСТЕЙ GIF - GIF CAPABILITIES ENQUIRY

Последовательность GCE идет из головного процесса и требует,

чтобы интерактивный декодер GIF вернул ответное сообщение, которое

определяет графические параметры для декодирования. Оно включает

возвращаемую информацию о доступных размерах экрана, числе битов

на цвет и поддерживаемом количестве цветов. Esc-последовательность

для GCE определена следующим образом:

ESC [ > 0 g (g в нижнем регистре, пробелы вставлены для

ясности)

(0x1B 0x5B 0x3E 0x30 0x67)

СООБЩЕНИЕ ВОЗМОЖНОСТЕЙ GIF - GIF CAPABILITIES RESPONSE

Ответное сообщение о возможностях GIF возвращается

интерактивным декодером и определяет возможности дисплея декодера

для всех графических режимов, поддерживаемых математическим

обеспечением. Заметьте, что оно может также включать графический

принтер, а не только экран монитора. Общий формат этого сообщения:

#version;protocol{;dev,width,height,color-bits,color-res}... <CR>

'#' - GCR символ-идентификатор (Знак номера)

version - номер версии формата GIF; начально '87a'

protocol='0' - Протокол end-to-end не поддерживается декодером.

Передача данных ведется непосредственным 8-битным

потоком.

protocol='1' - Может поддерживать протокол коррекции ошибок при

передаче данных от прямого хозяина на дисплей.

dev = '0' - Далее следуют параметры экрана

dev = '1' - Далее следуют параметры принтера

width - Максимальная ширина дисплея в пикселах

height - Максимальная высота дисплея в пикселах

color-bits - Поддерживаемое число битов на пиксел.

Следовательно, поддерживаемое число цветов

2\*\*color-bits.

color-res - Число битов на компоненту цвета, поддерживаемое

аппаратной цветовой палитрой. Если color-res равен

'0', таблица аппаратной палитры недоступна.

Заметьте, что все значения в GCR возвращаются в десятичных

числах ASCII и сообщение заканчивается символом "Возврат каретки".

- 11 -

Следующее GCR-сообщение описывает три стандартных режима

EGA с конфигурацией без принтера, поток данных GIF может

обрабатываться в рамках протокола с коррекцией ошибок:

#87a;1 ;0,320,200,4,0 ;0,640,200,2,2 ;0,640,350,4,2<CR>

ВВОД ГРАФИЧЕСКОГО РЕЖИМА GIF

Две последовательности, определенные ниже вызывают для работы

интерактивный декодер GIF. Между ними существует только

единственное отличие. Оно заключается в выборе различной среды

вывода. Эти последовательности:

ESC [ > 1 g Высветить изображение GIF на экране

(0x1B 0x5B 0x3E 0x31 0x67)

ESC [ > 2 g Выдать изображение непосредственно на

присоединенный графический принтер. Допускается

также необязательный вывод на экран.

(0x1B 0x5B 0x3E 0x32 0x67)

Заметьте, что символ 'g', заканчивающий каждую

последовательность находится в нижнем регистре.

ИНТЕРАКТИВНАЯ СРЕДА

Подразумеваемой средой при пересылке данных об изображении

GIF в интерактивных приложения является полностью 8-битный поток

данных от "хозяина" к получателю. Об установке 8-битного способа

пересылки данных при связи обычно должна заботиться головная

прикладная программа. Однако программа-получатель, поддерживающая

декодер GIF в линии связи, должна быть способна принимать и

передавать декодеру GIF все 256 возможных кодов 8-битных данных.

- 12 -

ПРИЛОЖЕНИЕ C

СЖАТИЕ И УПАКОВКА ИЗОБРАЖЕНИЯ

Поток растровых данных, которые описывают действительное

выходное изображение может быть представлен в следующем виде:

7 6 5 4 3 2 1 0

┌───────────────┐

│ код размера │

├───────────────┤ ───┐

│ байт-счетчик │ │

│ блока │ │

├───────────────┤ │

│ │ ├── Повторяется столько раз, сколько

│ байт данных │ │ необходимо

│ │ │

├───────────────┤ ───┘

. . . . . .

├───────────────┤

│0 0 0 0 0 0 0 0│ нулевой байт-счетчик

└───────────────┘ (заканчивает поток данных)

Преобразование изображения из серии значений пикселов к

передаваемому или запоминаемому потоку символов включает несколько

шагов. Вкратце эти шаги состоят в следующем:

1. Установка кода размера - Определяет число битов, необходимое

для представления действительных данных.

2. Сжатие данных - Сжатие серии пикселов изображения в серию

кодов сжатия.

3. Построение серии байтов - берет серию кодов сжатия и

преобразует их в строку 8-битных данных.

4. Упаковка байтов - Упаковка набора байтов в блоки, которым

предшествует символ-счетчик и вывод.

УСТАНОВКА КОДА РАЗМЕРА

Первый байт в потоке растровых данных GIF имеет значение,

указывающее минимальное число битов, необходимое для представления

для представления действительных значений пикселов. Как правило

оно будет таким же, что и число битов цвета. Однако из-за

некоторых ограничений алгоритма черно-белые изображения, которые

имеют один бит цвета, должны иметь код размера, равный 2. Такое

значение кода размера подразумевает также, что коды сжатия должны

быть на один бит длиннее.

СЖАТИЕ

Алгоритм LZW преобразует серию значений данных в серию кодов,

которые могут быть самими значениями или кодами, описывающими

серию значений. Если использовать аналогию с текстовыми символами,

то выходные коды состоят из символов и кодов, которые описывают

цепочки символов.

- 13 -

LZW-алгоритм, использованный в GIF алгоритмически

соответствует стандартному алгоритму LZW со следующими отличиями:

1. Определен специальный код очистки, который сбрасывает все

параметры сжатия/раскрытия и таблицы в исходное состояние.

Значение этого кода равно 2\*\*<код размера>. Например, если код

размера равен 4 (изображение имеет 4 бита на пиксел), код

очистки равен 16 (двоичное 10000). Код очистки может

появляться в любом месте потока данных и, следовательно,

требуется, чтобы LZW-алгоритм обрабатывал последующие коды

так, как будто бы начался новый поток данных. Кодировщик

должен выводить код очистки в качестве первого кода в каждом

потоке данных изображения.

2. Определен код конца информации, который явно указывает на

конец потока данных изображения. Если встретится такой код,

LZW-обработка прекращается. Этот код должен быть последним

кодом, формируемым кодировщиком для изображения. Значение

этого кода равно <Код\_очистки>+1.

3. Значение первого доступного кода сжатия равно <Код\_очистки>+2.

4. Выходные коды имеют переменную длину, начиная от

<код\_размера>+1 битов на код, до 12 битов на код. Тем самым

максимальное значение кода определяется равным 4095

(шестнадцатиричное FFF). Как только значение LZW-кода может

превысить текущую длину кода, длина кода увеличивается на

единицу. Паковщик и распаковщик этих кодов должны изменяться,

чтобы соответствовать новой длине кода.

ПОСТРОЕНИЕ 8-БИТНЫХ БАЙТОВ

Поскольку LZW-сжатие, используемое для GIF, создает серию

кодов переменной длины от 3 до 12 символов каждый, эти коды должны

быть переформированы в серию 8-битный байтов так, чтобы на самом

деле происходило запоминание или передача символов. Это

обеспечивает дополнительное сжатие изображения. Коды формируются в

поток битов так, как если бы они паковались справа налево, и затем

выбираются по 8 битов для вывода. Рассматриваемый массив 8-битных

символов при упаковке кодов длиной по 5 битов должен быть похож на

следующий пример:

байт n байт 5 байт 4 байт 3 байт 2 байт 1

┌─.....─────+────────+────────+────────+────────+────────┐

│ and so on │hhhhhggg│ggfffffe│eeeedddd│dcccccbb│bbbaaaaa│

└─.....─────+────────+────────+────────+────────+────────┘

Заметьте, что механизм физической упаковки будет изменяться

по мере того, как изменяется число битов в коде сжатия, но

концептуально он остается тем же самым.

- 14 -

УПАКОВКА БАЙТОВ

Как только байты созданы, они группируются в блоки для

вывода, причем каждому блоку предшествует байт-счетчик со

значением от 0 до 255. Блок с нулевым байтом-счетчиком заканчивает

поток данных для данного изображения. Эти блоки являются тем, что

выводится на самом деле в формате GIF. Такой формат блока

обеспечивает дополнительную эффективность за счет того, что

позволяет декодировщику считывать данные по мере необходимости,

читая сначала байт-счетчик, а затем пропуская сами данные об

изображении.

- 15 -

ПРИЛОЖЕНИЕ D

ОБРАБОТКА НЕСКОЛЬКИХ ИЗОБРАЖЕНИЙ

Поскольку поток данных GIF может содержать несколько

изображений, необходимо описать обработку и высвечивание таких

файлов. Поскольку дескриптор изображения допускает размещение

изображения в пределах логического экрана, можно определить

последовательность изображений, каждое из которых занимает часть

экрана, но их совокупность заполняет экран целиком. В подобных

ситуациях линии поведения при обработке изображений состоит в

следующем:

1. Не делать пауз между изображениями. Каждое обрабатывается

сразу же, как только будет распознано декодировщиком.

2. Каждое изображение переписывает любое другое изображение уже

находящееся внутри его окна. Экран очищается только в начале и

в конце обработки GIF-изображений. См. обсуждение терминатора

GIF.

Коды расширения:

0x1 – расширение простого текста (plain text)

0xF9 – расширение управления графикой (graphics control)

0xFE – расширение комментария (comment)

0xFF – расширение программы (application extension)

примеры:

[FF] — код расширения. В нашем случае имеем расширение программы.

[0B] — размер последующего блока в байтах.

[4E 45 54 53 43 41 50 45] — (NETSCAPE) идентификатор приложения, которому принадлежит это расширение.

[32 2E 30] — (2.0) код приложения. С его помощью приложение проверяет, действительно ли это расширение принадлежит ему.

[03] — размер последующего блока в байтах.

[01] — фиксированное значение.

[00 00] — значение 0..65535. Беззнаковое целое в формате little-endian. Определяет, сколько раз должен повторяться цикл. Для 0 – бесконечно.

[00] — конец блока.

[F9] — код расширения (расширение управления графикой).

[04] — размер последующего блока в байтах.

[04] –

0-2 биты

(000) – зарезервировано. Рекомендуется заполнять нулями.

3-5 биты

(001) — метод обработки. Определяет, что делать после отображения.

0 – к картинке не будет применяться никакой обработки

1 – картинка останется без изменений

2 – картинка затрется фоном

3 – восстановится изображение под картинкой

4-7 – не определены

6 бит

(0) – флаг ввода пользователя. Если 1, то для продолжения обработки изображения требуется реакция пользователя.

7 бит

(0) – флаг цвета прозрачности. Указывает, будет ли какой-нибудь цвет использоваться как прозрачный.

[32 00] – время задержки в анимации. = 50/100 секунды = 0.5 с

[00] – индекс цвета прозрачности.

[00] — конец блока.

**#pragma pack(push, 1)**

**/\* GIF file header structure \*/**

**struct HEADER**

**{**

**CHAR Sign[6]; // GIF file signature 'GIF8?a'**

**WORD W, H; // Image size**

**BYTE Info; // Image information byte:**

**// 0..2 bits - bits per pixel (- 1)**

**// 4..6 bits - color resolution (- 1)**

**// 7 bit - global color table presence flag**

**BYTE BackgroundColor; // Backgound color index**

**BYTE Reserved; // Not used (0)**

**}; /\* End of 'HEADER' structure \*/**

**/\* GIF image header structure \*/**

**struct IMAGE\_HEADER**

**{**

**WORD X0, Y0, W, H; // Image start coordinates and size**

**BYTE Info; // Image information byte:**

**// 0..2 bits - bits per pixel (- 1)**

**// 6 bit - interlacing flag**

**// 7 bit - local color table presence flag**

**}; /\* End of 'IMAGE\_HEADER' structure \*/**

**/\* GIF application extension (0xFF) structure \*/**

**struct EXT\_APPLICATION**

**{**

**CHAR Name[8]; // Application name**

**CHAR Version[3]; // Version info (for NETSCAPE: 0x01, 2bytes: 0..65535 repeat count)**

**}; /\* End of 'EXT\_APPLICATION' structure \*/**

**/\* GIF graphics extension (0xF9) structure \*/**

**struct EXT\_GRAPHICS**

**{**

**BYTE Info; // Graphics info bits:**

**// 0 bit: 1 if 'TransColor' used**

**// 1 bit: 1 if expected click between frames, 0 – auto**

**// 2-4 bits: disposal method:**

**// 0 – no deals**

**// 1 – image stay without changes**

**// 2 – image deleted by background**

**// 3 – restore image below frame**

**// 4-7 – not usage**

**WORD DelayTime; // Interframe delay time in 1/100 seconds**

**BYTE TransColor; // Transparent color index**

**}; /\* End of 'EXT\_GRAPHICS' structure \*/**

**#pragma pack(pop)**